Design Document

Fabio Gritti, Lorenzo Fontana , Riccardo Motta

January 31, 2014

Contents

1 Introduzione

Questo documento si pone l'obiettivo di fornire una descrizione dell'organizzazione ed del design dei dati tramite i modelli ER ed il modello logico tratto dall'ER. Inoltre tramite l'utilizzo di diagrammi UX l'obiettivo è di chiarificare le esperienze di navigazione dei vari attori durante l'utilizzo del sistema. Infine vengono partizionate le varie funzionalità tramite uno schema boundary control entity.

2 Design dei dati

2.1 Modello ER

Il modello ER rappresenta una vista concettuale di alto livello di come verranno memorizzate i dati gestiti dall'applicazione. Le entità possono essere suddivise in 3 diverse categorie:

1. *Entità legate alle informazioni degli utenti*: Utente,Utente Gruppo;
2. *Entità legate ai componenti e ai loro aggregati*: Trasporto, Volo, Hotel, Escursione, Viaggio, HotelPagato, EscursionePagata, VoloPagato, Pacchetto;
3. *Entità legate alle operazioni dell'utente*:Prenotazione, Invito, Gift List;
4. *Entità accessorie:* Amico, EscursionePagata;

2.1.1 *Informazioni degli utenti*

L'entity Utente contiene informazioni che verranno utilizzate come credenziali per effettuare il login (Username e Password), più dei dati anagrafici utili per identificare la persona.Insieme alla tabella Utente Gruppo sara possibile discriminare il ruolo della persona all'interno del sistema(Amministratore, Impiegato, Utente).

2.1.2 *Componenti e i loro aggregati*

L'entità trasporto è stata creata dal punto di vista logico per garantire l'estendibilità del sistema ad altri mezzi di trasporto nonostante, nella situazione attuale, si concretizzi soltanto nell'entità volo.

Nell'entità Hotel non è stato preso in considerazione il numero di posti disponibili. Questo perchè non è stato ritenuto di pertinenza dell'agenzia di viaggi, bensì dei singoli hotel che comunicheranno le date di disponibilità.

L'entità Viaggio tiene in memoria le informazioni (data inizio, data fine, hotel, voli andata e ritorno ed escursioni) dei viaggi che appartengono a prenotazioni, gift list o inviti e che pertanto devono essere salvate in maniera persistente.

Sono state poi create le entità VoloSalvato, HotelSalvato EscursioneSalvata il cui scopo è quelle di tenere le informazioni legate alle prenotazioni che pertanto non devono essere modificate da eventuali cambiamenti da parte degli impiegati che modificheranno invece le tabelle Volo Hotel Escursione.

2.1.3 *Operazioni dell'utente*

L'entità Prenotazione contiene le informazioni riguardanti l'acquisto di un viaggio, pertanto possiede una relazione a un Utente registrato e a un Viaggio.

L'entità Invito memorizza invece i dati relativi alla condivisione di un viaggio da parte di un utente. In particolare l'attributo mail amico serve a identificare con chi il viaggio viene condiviso mentre status permette di conoscere se la richiesta è stata accettata o è ancora in attesa.

L'entità Gift List è caratterizzata da due relazioni verso un solo Viaggio e un solo Utente registrato che permettono di identificarla univocamente. L'insieme di persone notificate della presenza della gift list è definito con l'attributo multiplo Amici che contiene un'insieme di mail. Infine lo status di pagamento dei vari componenti del viaggio viene definito con gli attributi VoloAPag VoloRPag HotelPag e l'attributo multiplo EscursioniPag.

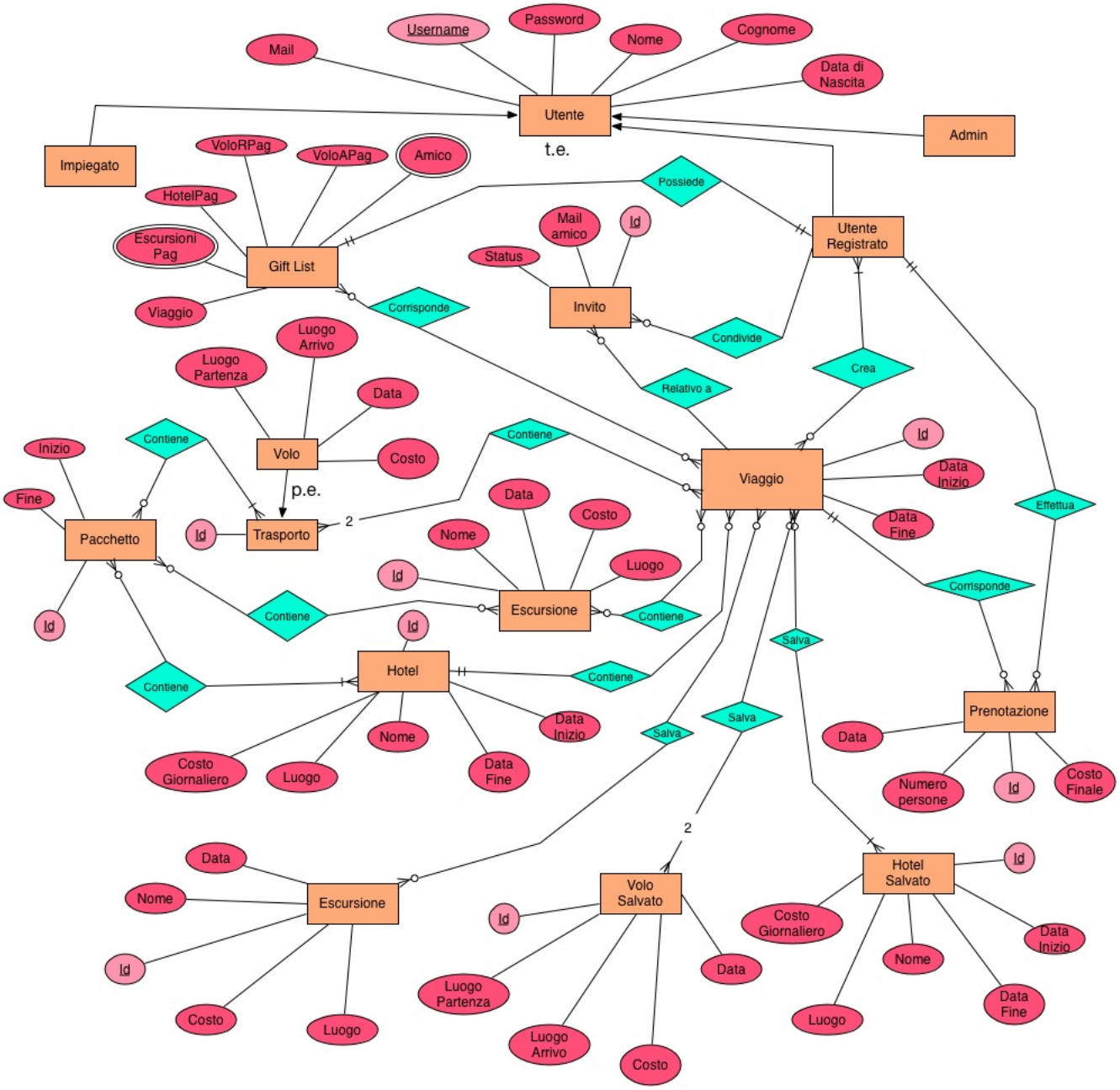


Figure 1: Modello ER

2.2 Schema Logico

Per passare dallo schema ER alla rappresentazione effettiva delle tabelle del database è necessario risolvere le gerarchie ISA, le relazione molti a molti e gli attributi multipli.

La gerarchia Utente viene risolta creando una tabella Utente, la quale include le varie informazioni anagrafiche e di identificazione, e una tabella UtenteGruppo, collegata alla prima per mezzo della chiave "username" e contenente i dettagli relativi al ruolo di ciascun iscritto.

Analogamente la gerarchia concettuale Trasporto viene risolta creando una tabella Volo in quanto, considerando in futuro l'inserimento di altri mezzi di trasporto, questi avranno attributi molto diversi tra loro rendendo così necessaria la creazione di diverse tabelle.

Le relazione molti a molti vengono risolte introducendo una tabella intermedia le cui chiavi primarie sono foreign key delle tabelle che voglio legare. Ad esempio tra Hotel e Pacchetto verrà creata una tabella HotelPacchetto con due attributi Hotel_ID e Pacchetto_ID che sono sia chiavi primarie di ViaggiPacchetto sia foreign key per le 3 rispettive tabelle Hotel e Pacchetto.

Infine, nel modello logico, l'attributo multiplo amico, appartenente a Gift List, si realizza con una tabella Amico che conterrà delle mail e viene legata a Gift List con tabella intermedia analoga al caso precedente.

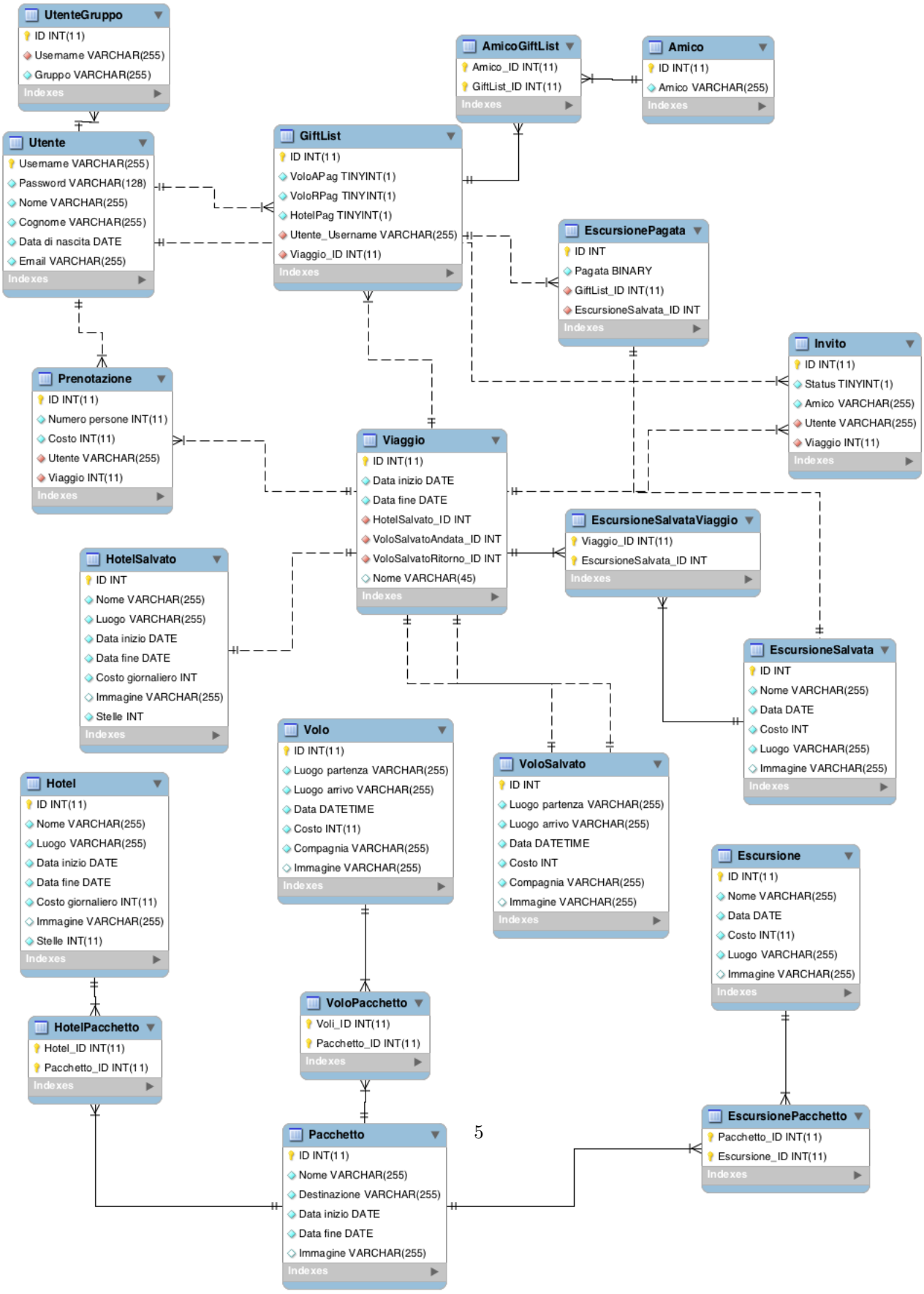


Figure 2: Schema Logico

3 Design dell'applicazione

3.1 Livelli Architetturali

La piattaforma software scelta per la realizzazione dell'applicazione web "TravelDream" è Java Enterprise Edition, con l'aggiunta dell' application server GlassFish, il quale permette la gestione del web server e l'utilizzo di tecnologie quali le Java Server Faces e le EJB (Enterprise JavaBeans). Per garantire un elevato grado di flessibilità, il sistema TravelDream si andrà a mappare su una tipica architettura a più livelli:

1. Client Tier: è il tier che permette all'utente di interagire con la web application è costituito da un web browser in grado di interpretare pagine HTML5, stylesheet CSS3 e Javascript;
2. Web Tier: è il tier che si occuperà di gestire le richieste http provenienti dal browser web, di interagire con il Business tier per operare sui dati del sistema e di generare la pagine di risposta per l'utente. Sarà composto da componenti web quali JSF;
3. Business tier: è il tier che consentirà la creazione, l'interrogazione e la modifica dei dati. Si occuperà inoltre di garantire la persistenza dei dati. Sarà composto da Entity Java Bean e Session Bean che gestiranno la logica;
4. Data Tier: costituito da un database relazionale, gestito da un server MySQL.

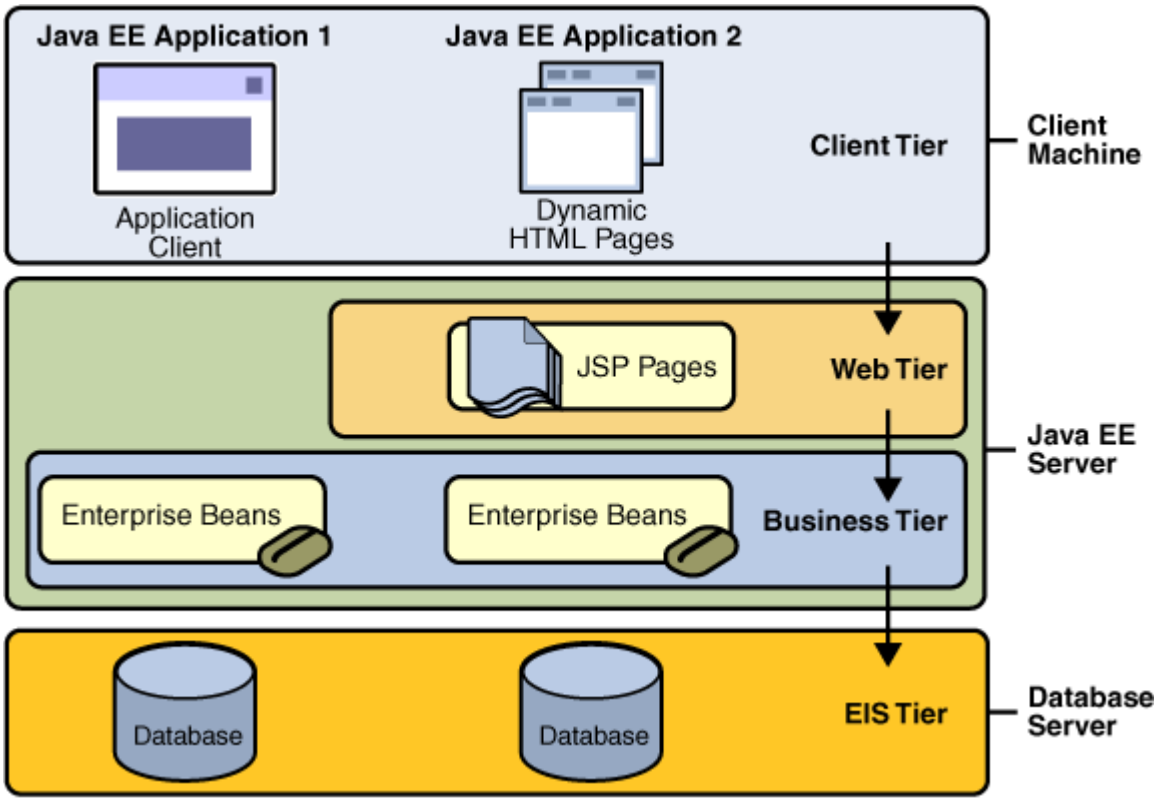


Figure 3: Multi-tier

4 Modelli di navigazione

In questa sezione verranno presentate le esperienze di navigazione di tutti gli attori e il flusso relativo alla creazione di una condivisione. Vengono utilizzati dei diagrammi UX che schematizzano le diverse pagine incontrate durante l'utilizzo del sistema. Per ogni diagramma verrà presentata una breve descrizione. Le situazioni di errore sono state inserite solo in alcune circostanze per evitare di appesantire gli schemi. Le pagine marcate con il simbolo $ hanno un doppio significato:

- da qualsiasi sotto-pagina possiamo ricondurci a queste in ogni momento;
- queste pagine possono essere utilizzate indiscriminatamente come punto di inzio navigazione.

4.1 UX login e registrazione

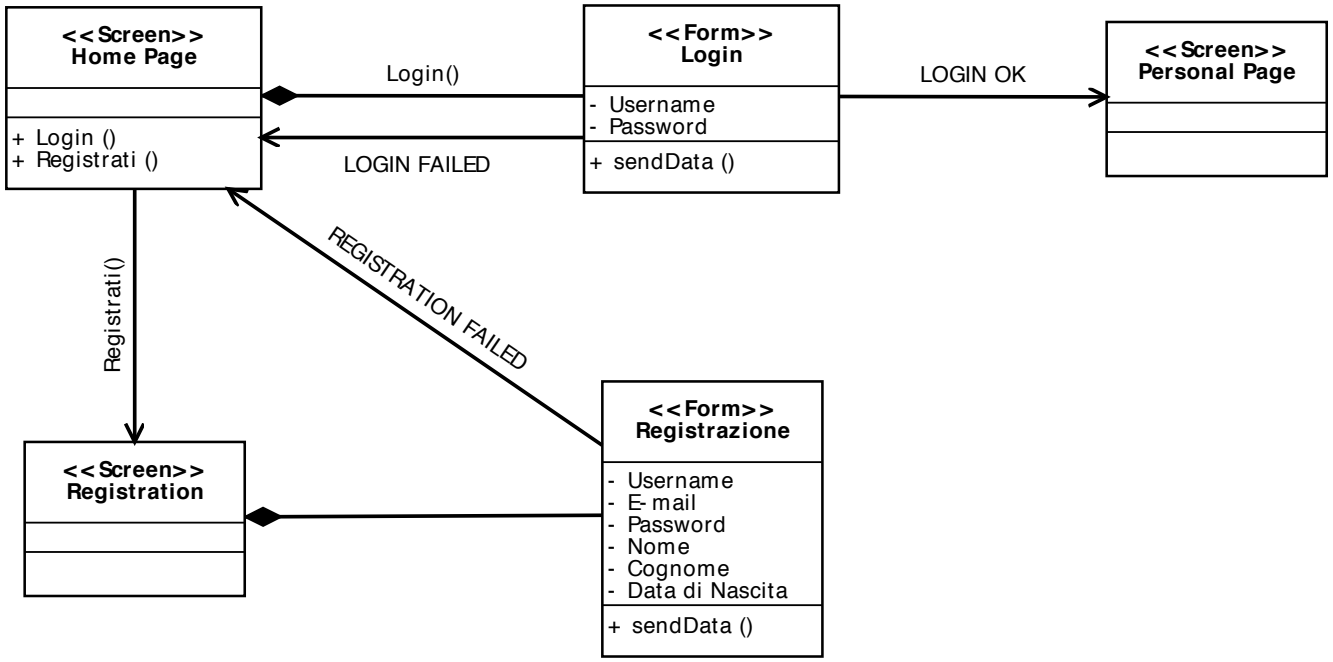


Figure 4: UX login e registrazione

Lo schema sopra riportato viene utilizzato anche per rappresentare il login dell'impiegato nel sistema; l'unica differenza sta nel fatto che non è prevista una schermata di registrazione per esso, dato che sarà l'admin ad inserire impiegati nel sistema. Le screen principali sono essenzialmente due, la prima permette di inserire le proprie credenziali se si possiede già un account, mentre la seconda permette di registrarsi. Se la fase di login va a buon fine, l'utente registrato viene reindirizzato alla sua home page utente; tale pagina è ben definita nel successivo schema UX.

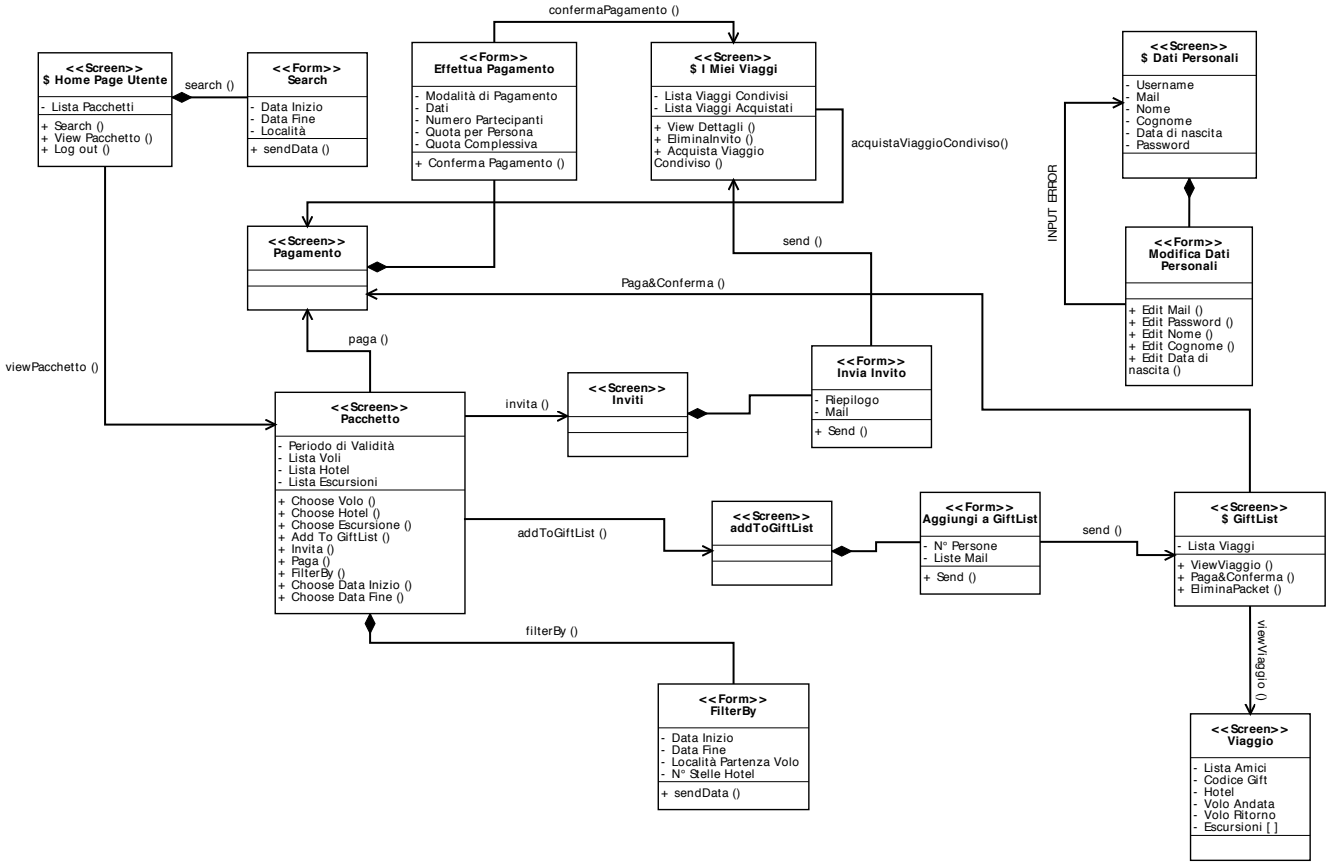


Figure 5: UX utente registrato

4.2 UX utente registrato

Lo schema rappresenta il modello di navigazione degli utenti registrati all'interno del sistema, la navigazione parte dalla HomePageUtente. Ecco una breve descrizione delle principali view incontrate:

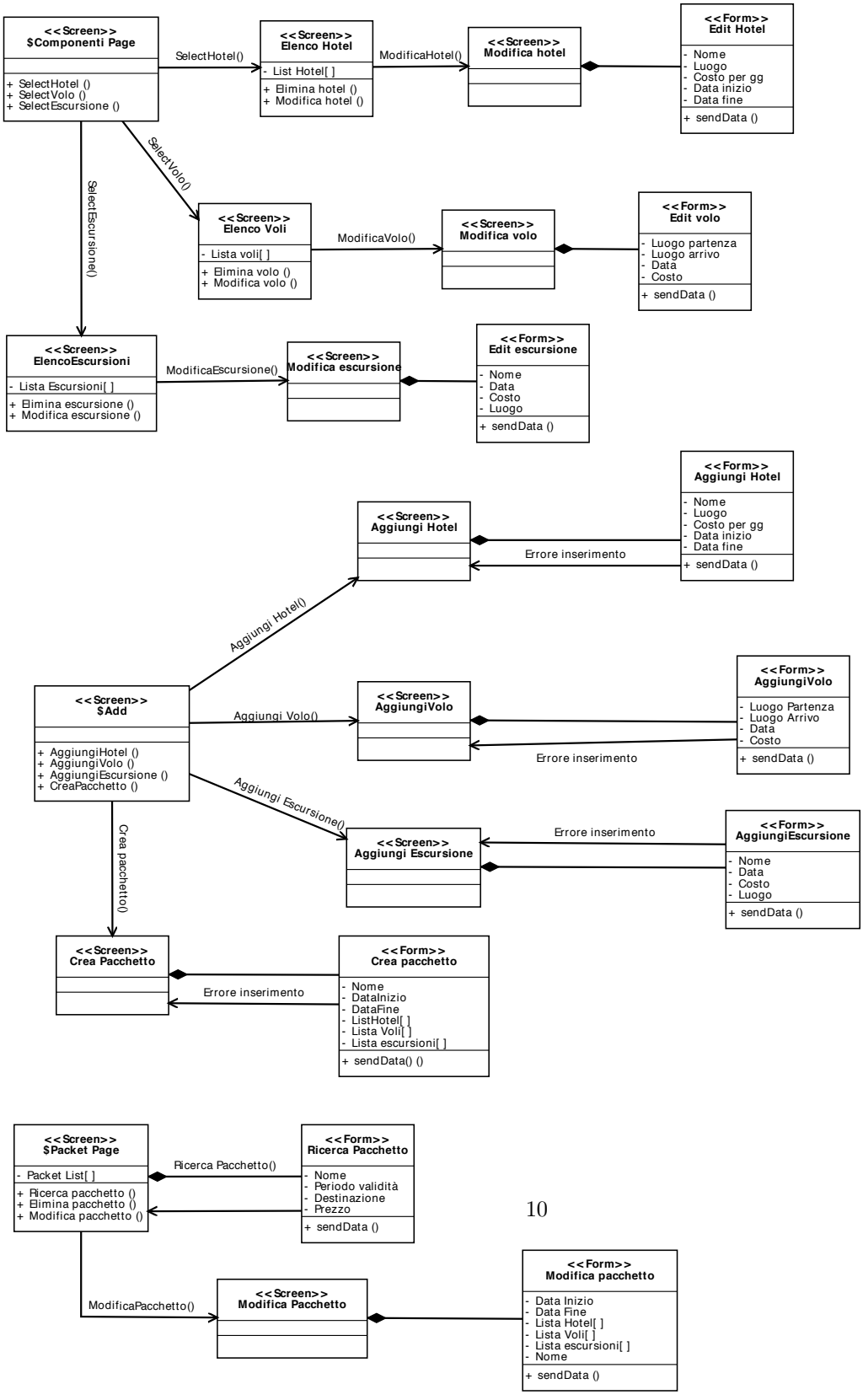
1. *Home Page Utente*: è la pagina visualizzata dall'utente dopo aver effettuato il login, contiene una lista di pacchetti e gli strumenti per filtrarli in base a diverse caratteristiche. Da questa pagina è possibile selezionare un pacchetto per visualizzarne i componenti offerti;

2. *Pacchetto*: In questa pagina vengono visualizzati tutti gli elementi che sono disponibili all'interno del pacchetto.Da qua è possibile selezionare le date di partenza e arrivo ed aprire le pagine di editing dei componenti: hotel , volo, escursioni (se presenti). Inoltre una volta creato il viaggio in questa screen, è possibile condividerlo con un amico, aggiungerlo alla gift list, oppure procedere all'acquisto. Si noti che da questa pagina si procede alla creazione di un viaggio a partire da elementi messi a disposizione in quel pacchetto, la finaliz-

zazione dell'operazione prevede un inserimento all'interno della gift list del viaggio creato, un invito per quel viaggio ad altri amici, oppure un acquisto con prenotazione del medesimo;

3. *I MieiViaggi*: In questa pagina vengono visualizzati i viaggi condivisi con altri amici ed i viaggi acquistati; è la pagina in cui si arriva dopo aver acquistato un viaggio o mandato un invito di partecipazione ad un amico;
4. *GiftList*: Pagina che mostra tutti i viaggi presenti nella gift list, presentando un breve riepilogo per ogni viaggio (cosa è stato pagato, chi ha pagato cosa , ecc...) . Da questa schermata è possibile procedere al pagamento di uno dei viaggi salvati;
5. *addToGiftList*: contiene una form per l'inserimento di una personale descrizione del viaggio e le mail degli amici a cui notificarlo;
6. *Inviti*: Screen che permette all'utente di mandare un invito ad un amico, previa inserimento dell'indirizzo mail dello stesso;
7. *Pagamento*: pagina che consente il pagamento di un viaggio, contiene tutto il necessario per inserire i dati di per l'acquisto;
8. *Dati personali*: Pagina che permette di modificare alcuni dati personali dell'utente.

4.3 UX impiegato



Lo schema rappresenta il flusso delle pagine navigate dall'impiegato durante l'esecuzione del proprio lavoro. Segue una breve descrizione:

1. *Componenti page*: pagina in cui vengono ricercati/modificati/eliminati i componenti base (hotel , voli , escursioni);
2. *Add page*: pagina in cui si possono aggiungere nuovi elementi base al sistema oppure si può procedere alla creazione di un nuovo pacchetto;
3. *Packet page*: pagina in cui è possibile ricercare, modificare o eliminare interi pacchetti precedentemente creati.

4.4 UX admin

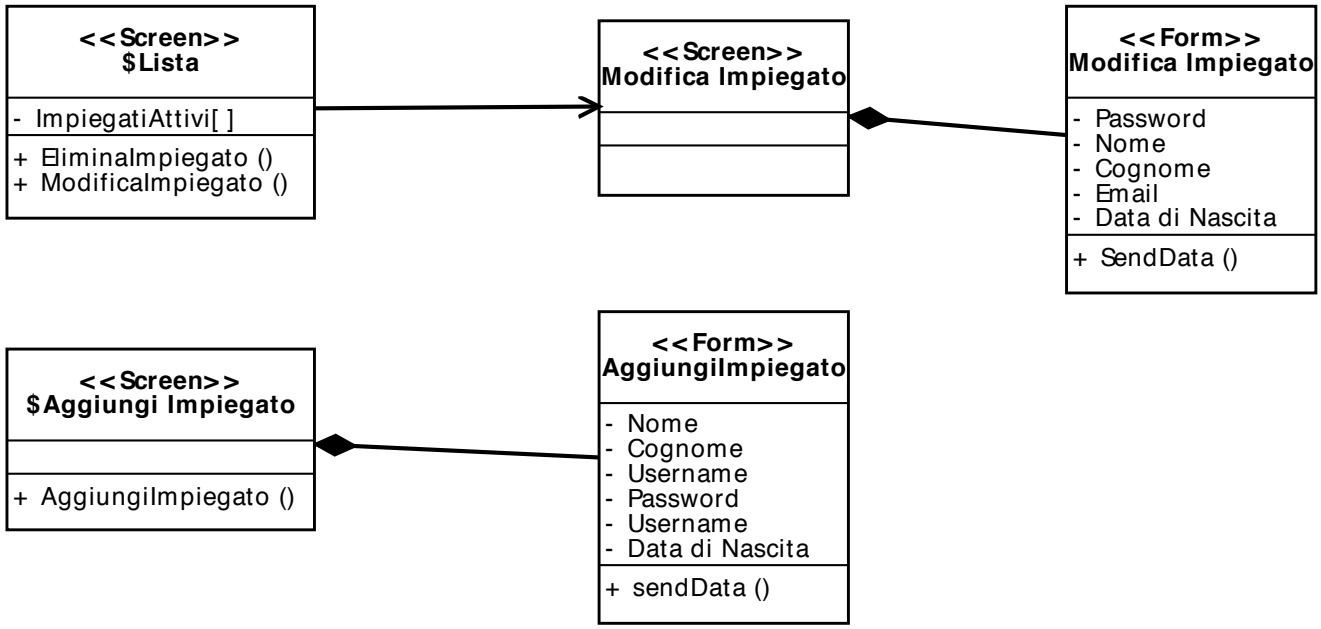


Figure 7: UX admin

L'admin dispone di due pagine di servizio: una mostra l'elenco degli impiegati aggiunti nel sistema, l'altra permette l'aggiunta di un nuovo impiegato.

4.5 UX condivisione

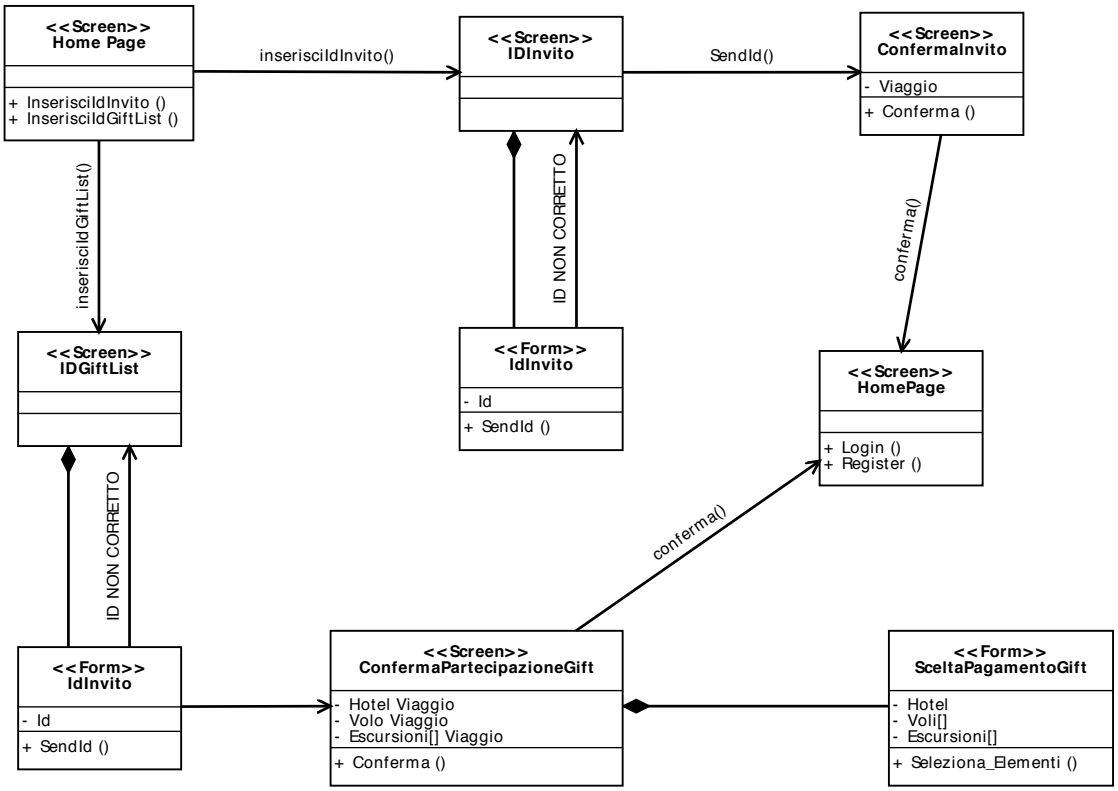


Figure 8: UX condivisione

Dopo aver ricevuto un link via mail (relativo ad un invito od alla partecipazione ad una gift list) si arriva sulla home page dove si hanno due opzioni: la prima è quella di inserire l'ID di un viaggio a cui si è stati invitati a partecipare, l'altra è quella di inserire un ID per una gift list. Seguendo la prima opzione si viene reindirizzati su una pagina che offre un riepilogo del viaggio; a questo punto si può decidere di non confermare la partecipazione oppure di confermarla loggandosi nel sistema (se registrati) , o registrandosi al sistema se non si possiede ancora un account. La seconda opzione disponibile è quella di inserire un ID per una gift list , fatto questo verrà proposta una schermata in cui sarà possibile selezionare o meno quello che si vuole pagare relativamente a quel viaggio, una volta fatte le scelte per finalizzare l'operazione occorre loggarsi al sistema con un account esistente, oppure registrare un nuovo account. Le screen di Registrazione e Login sono già state presentate negli UX precedenti.

5 Boundary Diagram

Lo schema architetturale dell'applicazione risulta molto legato al pattern model view controller che JavaEE supporta con la sua struttura, pertanto l'analisi dell'implementazione è stata realizzata con uno schema Boundary Control Entity.

I Boundary rappresentano la logica di presentazione, ovvero le pagine xhtml che verranno sviluppate usando la tecnologia JSF. Per rendere più compatto lo schema in un'unica Boundary sono state fuse diverse pagine caratterizzate da funzionalità simili così da concentrare l'attenzione sulla interazione tra Boundary e Control. Perciò nei Boundary sono stati riportati solo quei metodi che avranno un interazione con il Control, in questo modo metodi come la selezione dell'aereo nella fase di creazione del viaggio sono stati trascurati in quanto queste informazioni verranno salvate temporaneamente in un Managed Bean.

Il Control rappresenta invece la logica di Business dell'applicazione che verrà realizzata attraverso i Session Beans.L'approccio scelto in questo caso è stata la divisione rispetto alle funzionalità che i Bean devono svolgere.Ad esempio è stato creato il Control Gestione Utente chiamato da molte Boundary che permetterà di gestire il profilo dei diversi attori del sitema (Impiegati e Utenti).

Infine le Entity rappresentano le tabelle del database su cui i Control interagiranno e pertanto coincidono con alcune delle tabelle descritte nello schema logico (chiaramente nelle entity non verranno rappresentate le tabelle ponte necessarie per le relazioni molti a molti).

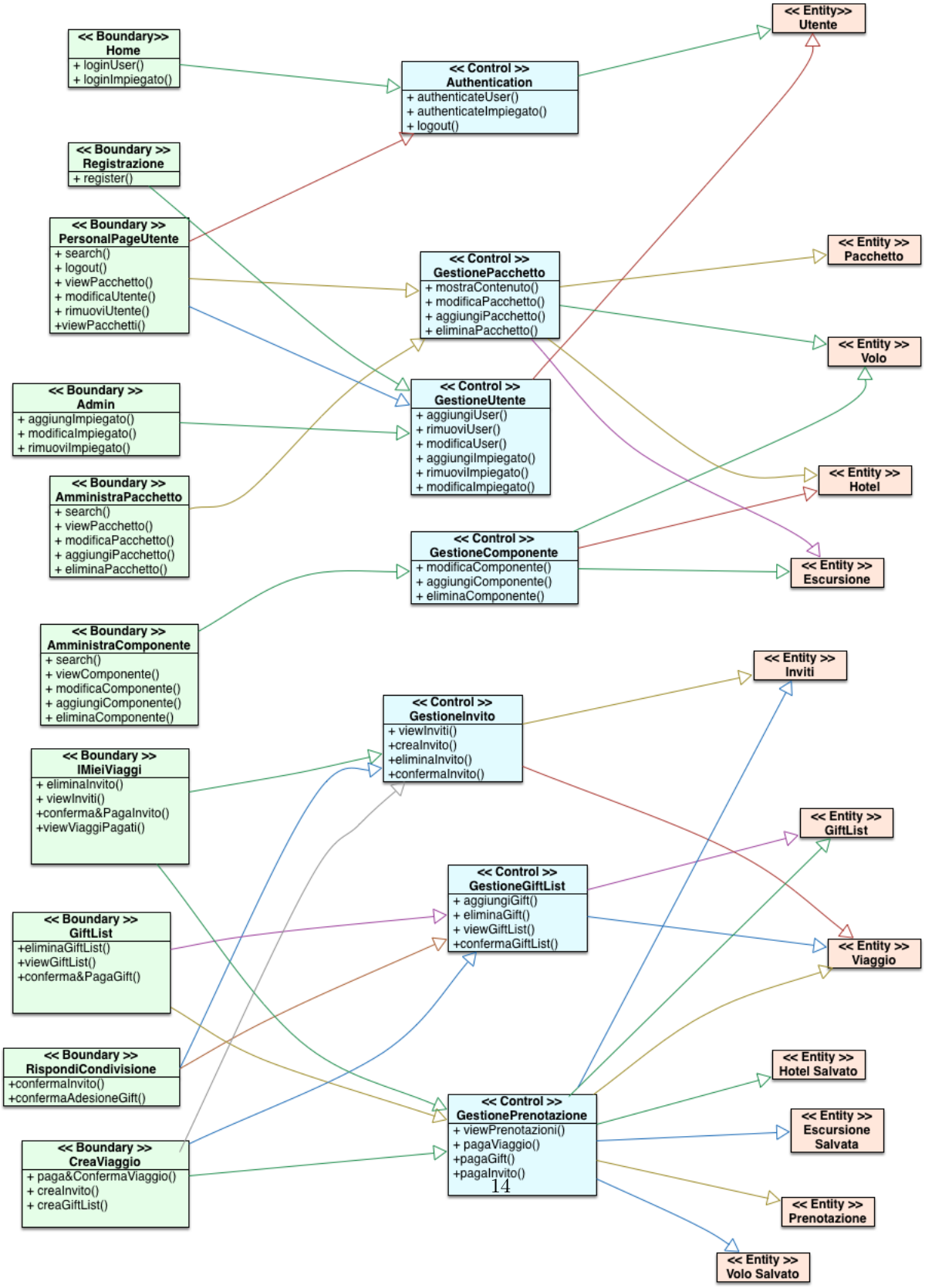


Figure 9: Boundary diagram

6 Diagrammi di sequenza

6.1 Diagramma “invito”

Il primo Sequence Diagram da noi proposto riguarda la creazione di un viaggio e la sua condivisione con un amico. Per prima cosa l’utente utilizza la funzione loginUser() dalla << Boundary >> “Home”, la quale reperisce le informazioni necessarie al login tramite il Controller Authentication. A questo punto, dalla << Boundary >> “PersonalPageUtente” viene effettuata una search(), fatta filtrando i pacchetti in base ad alcune caratteristiche. La Boundary utilizza il << Control >> “Search” per ricercare le << Entity >> “Pacchetto” che soddisfano i criteri di ricerca. Successivamente l’utente seleziona uno dei pacchetti tramite la viewPacchetto() chiamata sul boundary che reperisce il contenuto di tale pacchetto utilizzando il <<Control>> gestione pacchetto. L’utente può personalizzare ora il pacchetto andando a selezionare gli elementi di suo gradimento. Finita la personalizzazione si utilizza il creaInvito() della <<Boundary>> CreaViaggio per creare e condividere l’invito con un amico. Tutto questo comporta la creazione di una nuova entity invito all’interno della corrispondente tabella.

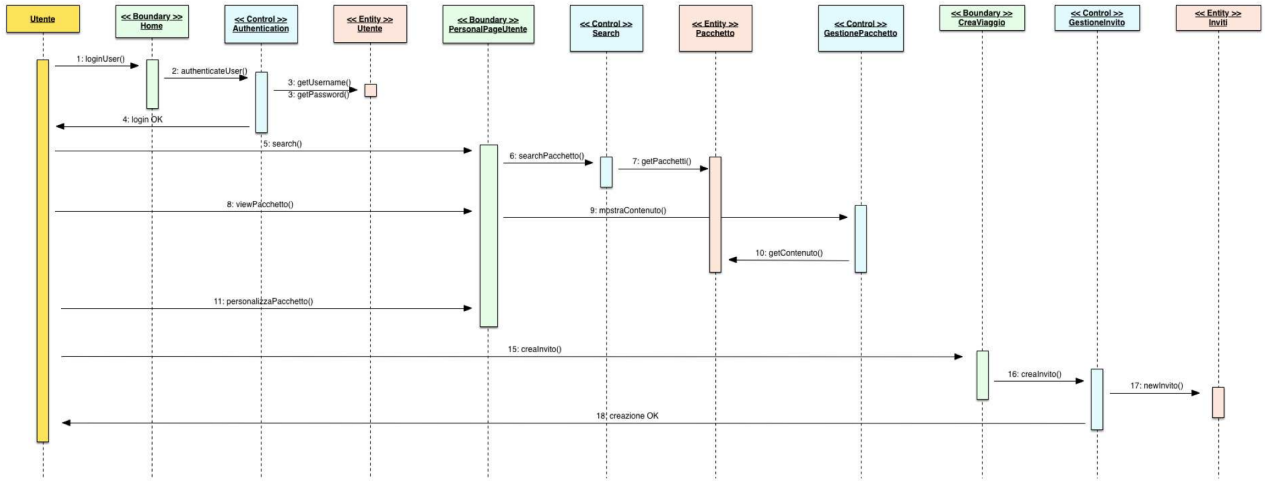


Figure 10: SD Invito

6.2 Diagramma “paga giftlist”

Il secondo sequence Diagram descrive le azioni che un utente non ancora registrato nel sistema deve compiere per poter pagare parte di una gift list. Dopo averla visualizzata, partendo dalla << Boundary >> “home”, all’utente è richiesta una registrazione. Questa è possibile sfruttando il << Control >> “Gestione Utente” che inserisce una nuova << Entity >> “Utente”. A questo punto non rimane altro che effettuare il pagamento: dal << Boundary >> “RispondiCondivisione” il << Control >> “Gestione Gift List” opera sulla << Entity >> “Gift List” in modo da aggiornarne i dati in base ai pagamenti effettuati.

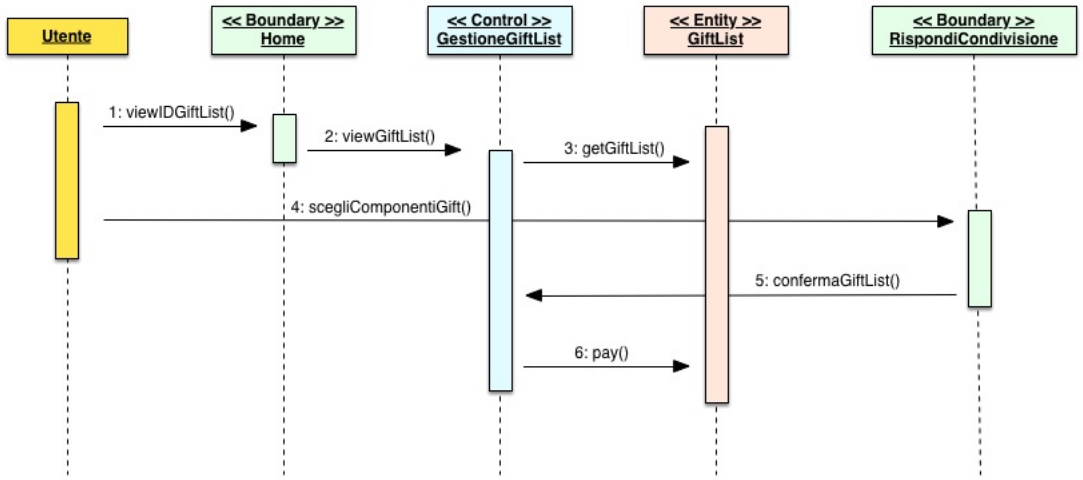


Figure 11: SD paga GiftList

6.3 Diagramma "aggiungi componente"

In quest'ultimo diagramma di sequenza andiamo a mostrare l'aggiunta di un componente (hotel) al database aziendale della TravelDream. Dopo essersi loggato, l'impiegato sarà in grado di svolgere la mansione precedentemente descritta grazie ad una semplice operazione sulla << Entity >> "Hotel", realizzata a partire dalla << Boundary >> "AmministraComponente" grazie al << Control >> "GestioneComponente".

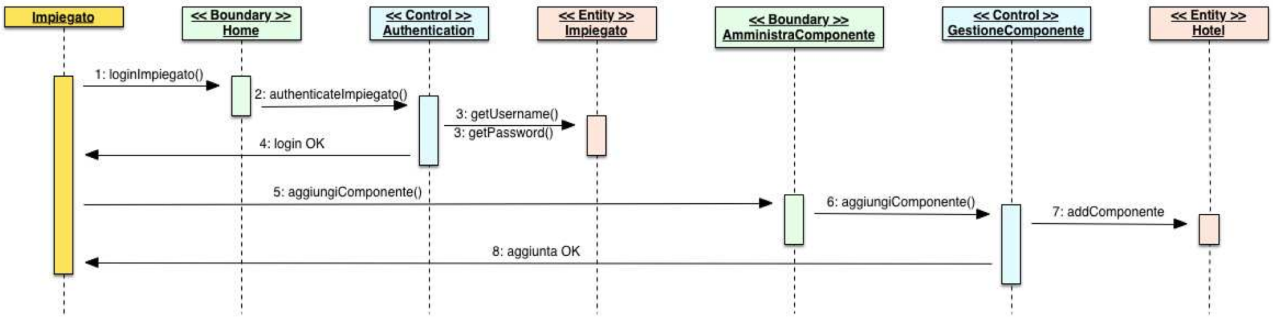


Figure 12: SD aggiungi componente